Information-Theoretic Security
Overview

• **Goal**: Obtain information-theoretic security against an **active** adversary.
  – Active adversary may arbitrarily modify corrupted players’ behavior.
  – Consequently, simulator must also play a more active role.

• **Realizable adversary structures**:
  – **Perfect security,** $Q^{(3)}$ structures ($t < n/3$)
    - $t < n/3$ necessary for realizing broadcast (or Byzantine Agreement)
    - $t < n/3$ optimal even when broadcast is assumed
  – **Statistical security,** $Q^{(2)}$ structures ($t < n/2$), assuming broadcast

• **Today**: Perfect security using CNF secret sharing
  – Pros: simple, general.
  – Con: inefficient when $n$ is large, even for threshold structures.

• **Not in this lecture**: Perfect security using Shamir
  – Efficient for threshold structures.
How to Flip a Coin?

- Nontrivial because of rushing
- Reduces to securely computing modular sum
  - Output = sum of randomly chosen inputs
- Variant of previous protocol for sum:
  - R1: share $x_i$ according to $L$
  - R2: add up the $n$ received shares and send the result $m_i$ to all players
  - output the sum of all $m_i$.
- Security problems
  - $P_i$ may send different $m_i$ to different players
    - use broadcast to ensure consistency
  - $P_i$ may choose $m_i$ after learning the other $m_j$
    - use redundant $L$ (e.g., $t$-private with $t < n/3$) to allow error-correction
    - works if players are honest during R1
    - what can be done against dishonest behavior during R1?
Verifiable Secret Sharing

• Verifiable Secret Sharing for $L$ is a $T$-secure protocol realizing the following functionality $\text{VSS}_L$:
  – $P_1$ (dealer) has inputs $s, \rho$; other players have no input.
  – Players’ outputs are $L(s, \rho)$
  – Intended use: dealer picks $\rho$ at random

• Features
  – Privacy: if dealer is uncorrupted and $\rho$ is picked at random, adversary learns nothing about $s$.
  – Consistency: outputs of uncorrupted players are consistent with $L(s^*, \rho^*)$ for some $s^*, \rho^*$
    • even if dealer is corrupted!
  – Correctness: if dealer is uncorrupted, then consistency holds with $s^* = s$. 

**T-Robustness**

- A secret-sharing scheme is **T-robust** if, after $s$ has been shared, a T-adversary cannot prevent the correct reconstruction of $s$.

![Diagram showing the secret-sharing process with referrer and players $P_1, P_2, P_3, P_4$.]

- **Adversary’s goal:** create ambiguity with nonzero probability
  - May assume that adversary knows all shares $s_i$
  - **Key point:** Referee does not know which players are corrupted

$s_i$ are consistent with $L(s,\rho)$
Robustness (contd.)

• Claim: a T-private L is T-robust iff T is of type Q(3).
  – Def. T is of type Q(3) if for each \( T_1, T_2, T_3 \in T \), \( T_1 \cup T_2 \cup T_3 \neq [n] \)
  – Threshold case: \( t < n/3 \)

• If:
  – Suppose towards contradiction that ambiguity is possible.
  – There is a corrupt share vector \( s' \) which is T-consistent with two valid share vectors \( s^1, s^2 \) corresponding to distinct secrets \( s^1, s^2 \)
  – Define \( T_1 = \{ i : s'_i \neq s^1_i \} \), \( T_2 = \{ i : s'_i \neq s^2_i \} \), \( T_3 = [n] \setminus (T_1 \cup T_2) \)

From the correctness of L, \( T_3 \) must be in T

\[
\begin{align*}
T_1 & = \{ \text{indices where } s'_i \neq s^1_i \} \\
T_2 & = \{ \text{indices where } s'_i \neq s^2_i \} \\
T_3 & = [n] \setminus (T_1 \cup T_2) \\
\end{align*}
\]
Robustness (contd.)

• **Claim:** A $T$-private $L$ is $T$-robust iff $T$ is of type $Q^{(3)}$.

• **Only if:**
  
  - Suppose there are $T_1, T_2, T_3 \in T$ such that $T_1 \cup T_2 \cup T_3 = [n]$  
    - Wlog $T_1, T_2, T_3$ are pairwise disjoint  
  - Let $s^1, s^2$ be distinct secrets, and $s^1, s^2$ corresponding share vectors which agree on all their $T_3$-entries.
    - Such $s^1, s^2$ exist by the privacy requirement (o/w $T_3$ could perfectly distinguish between $s^1, s^2$).
  - Define a corrupt share vector $s'$ which agrees with $s^1$ on $T_1$, with $s^2$ on $T_2$, and with both on $T_3$.
    - Adversary can turn valid shares into $s'$
    - $s'$ is $T$-consistent with both $s^1, s^2$!
Robustness (contd.)

• On the cost of reconstruction
  – Naïve approach: check all possible \((s, \rho)\)
  – Better approaches for useful schemes
    • Shamir’s scheme: use efficient decoding algorithm for Reed-Solomon codes.
    • CNF scheme: use majority vote to recover each additive share \(s_T\)
    • In fact, above schemes allow reconstruction of the original share vector (not only the secret).

• Significance of \(T\)-robustness
  – Suppose \(s\) has been shared using \(\text{VSS}_L\), where \(L\) is \(T\)-robust, and then each player sends its share to the referee.
    • Corrupt players may send incorrect shares
  – If dealer is honest, referee will correctly reconstruct \(s\).
  – Even when dealer is corrupted, at the end of the VSS it is effectively committed to unique secret \(s^*\) which it cannot change later.
Application: Simultaneous Broadcast

- \( \text{SB}(x_1, \ldots, x_n) = x_1 \circ x_2 \circ \ldots \circ x_n \)
- **Note:** Secure computation of \( \text{SB} \) \( \rightarrow \) Coin Flipping

- Fix some \( T \)-private \( L \) where \( T \) is of type \( Q^{(3)} \).

**Perfect \( T \)-secure reduction from \( \text{SB} \) to \( \text{VSS}_L \):**

- Each \( P_i \) picks random \( \rho_i \) and calls \( \text{VSS}(x_i, \rho_i) \) to share \( x_i \)
- Players exchange all shares
  - No need for broadcast!
- For each \( j \), \( P_i \) outputs the unique \( x_j^* \) which is \( T \)-consistent with its \( n \) shares.

**Q:** How can the protocol be generalized to compute an arbitrary linear func.?
Simultaneous Broadcast (contd.)

- **T-secure reduction from SB to VSS\(_L\):**
  - Each \( P_i \) calls VSS\((x_i, \rho_i)\) to share \( x_i \)
  - Players exchange all shares
  - For each \( j \), \( P_i \) outputs the unique \( x_j^* \) which is T-consistent with its \( n \) shares.

- **Simulating an active adversary \( A \) corrupting \( T \):**
  - Simulator’s input: \( x_T \)
    - Outputs are not yet available!
  - Simulator picks random \( \rho_T \), runs \( A \) on \((x_T, \rho_T)\), and “extracts” its effective inputs \( x_T^* \) from messages \( (x_i^*, \rho_i^*)\) it sends to VSS oracle.
    - If no message is sent, a default value is taken (same as in protocol).
  - Simulator sends \( x_T^* \) to TP, and receives an \( n \)-tuple of outputs \( y^* \).
  - Messages from VSS oracle and uncorrupted players are simulated as follows:
    - For each \( i \in T \), the corresponding \( n \)-tuple of messages is \( L(y_i^*, \rho_i^*)\)
    - For each \( i \notin T \), the corresponding \( n \)-tuple of messages is \( L(y_i^*, \rho_i) \), where \( \rho_i \) is random.
Composition Revisited

• Motivation
  – **Inwards**: supports modular design of a protocol using sub-protocols.
  – **Outwards**: guarantees that a protocol is safe to use as a building block in higher-level applications.

• Protocol composition in passive case is relatively simple.

• Active case is more problematic.
  – First observed in the context of Zero-Knowledge Proofs

• Major distinction between
  – Sequential composition: make one call to TP in each round
  – Universal composition: arbitrary calls to TP

• Our security definitions support* sequential composition
  * In active computational case, need to slightly strengthen definition by allowing adversary and simulator to be nonuniform (get an “advice”).

• To get universal composition, stronger definition is needed
  – **UC-security** allows environment to *interactively* distinguish between REAL and IDEAL.
  – Natural information-theoretic protocols are in fact UC-secure
CNF-based VSS

• Let $L$ be the $T$-private CNF scheme.
  – A secret $s \in F$ is shared as follows:
    • Additively share $s$ into $\sum s_H$, where $H$ ranges over all complements of maximal subsets in $T$.
    • Send each additive share $s_H$ to all players in $H$.

• Intuition for VSS
  – Difficulty: guarantee consistency even when dealer is corrupted
  – Key point: All players in $H$ can safely exchange their received values of $s_H$ over secure channels without violating privacy.
  – Whenever an honest player finds inconsistency, it can safely broadcast a complaint forcing dealer to broadcast $s_H$.
    • No privacy concern: adversary can cause complaints only if it corrupts the dealer or some player in $H$. In both cases it already knows $s_H$.
    • If dealer does not comply, it is “disqualified” and a default sharing is used (e.g., all shares are 0). Whether dealer is disqualified is public.
  – After all complaints are resolved, the shares held by honest players are consistent with a valid $L$-sharing of some secret $s^*$.
CNF-based VSS: The Protocol

• R1:
  – Using $s$, $\rho$, dealer creates additive sharing $s = \sum s_H$, where $H$ ranges over all complements of maximal subsets in $T$, and sends each additive share $s_H$ to all players in $H$.

• R2:
  – For each set $H$, each player in $H$ sends to all other players in $H$ its received value of $s_H$.

• R3:
  – If a player in $H$ observes inconsistent values of $s_H$, it broadcasts a complaint: “reveal $s_H$”.

• R4:
  – Dealer broadcasts all $s_H$ for which a complaint has been made.

• Output:
  – Each $P_i$ outputs all received $s_H$ such that $i \in H$, overriding original values with values broadcast by the dealer.
Robust Partial Functionalities

- Recall: partial functionality is defined over a subset $X$ of the inputs.
- How does one define a reduction to a partial functionality?
  - Active adversary may cause input sent to TP to be illegal.
- A partial functionality $g$ is $T$-robust if it is possible to (efficiently) determine $g(x)$ from a $T$-corrupted version of $x$.
  - If $L$ realizes a $Q^{(3)}$ structure $T$, then $MS_L$ is $T$-robust
- A $T$-secure reduction from $f$ to a $T$-robust $g$ is well defined.
  - TP returns the unique value of $g$ consistent with the given inputs.
Securely Reducing $f$ to VSS+MS

• Let $L$ be a linear secret-sharing scheme over $F$ realizing a $Q^{(3)}$ structure $T$, and let $MS=MS_L$.
  – $MS$ is $T$-robust
• Let $C$ be an arithmetic circuit over $F$ evaluating $f$.

- **Input sharing stage:**
  - Each player calls $VSS_L$ to share its inputs.
    • Now all inputs are “fixed”.
- **Computation stage:**
  - Suppose $a,b$ were already shared, and let $[a],[b]$ denote their share vectors.
    – $c=a+b$: let $[c]←[a]+[b]$ (no interaction)
    – $c=a*b$: let $[c]←MS([a],[b])$
- **Output reconstruction stage:**
  - For each output $z$, player exchange all shares $[z]$ and output the unique secret which is $T$-consistent with received shares.
Reducing $\text{MS}_{\text{CNF}}$ to $\text{VSS}_{\text{CNF}}$

- Let $L = T$-private CNF for some $Q^{(3)}$-structure $T$.
- **Inputs:** $L$-shares $[a],[b]$ of secrets $a,b$, where $a=\sum a_H$, $b=\sum b_H$, and each $a_H$, $b_H$ are included in the inputs of all players from $H$.
  - Sets $H$ are complements of sets in $T \Rightarrow$ each $H_1,H_2,H_3$ intersect
- **Outputs:** random CNF shares $[c]$ of $c = ab = \sum a_H b_H$.

**Sharing the terms**
- For each $H,H'$, players in $H \cap H'$ evaluate $m_{H,H'} = a_H b_H$, and VSS this value.
- Let $[m_{i,H,H'}]$ denote the shares originating from $P_i$.
  - **Note:** some $i \in H \cap H'$ must be honest. Otherwise, if $T \in T$ contained all players in $H \cap H'$, $H' = [n] \setminus T$ would satisfy $H \cap H' \cap H'' = \emptyset$.

**Revealing differences**
- Players reconstruct $m_{j,H,H'} - m_{j',H,H'}$ for each successive $j,j'$ in $H \cap H'$
  - Done by locally computing $[m_{j,H,H'} - m_{j',H,H'}]$ and exchanging shares.
  - If all $|H \cap H'| - 1$ differences are 0, all values $m_{i,H,H'}$ are good and players set $[m_{H,H'}] = [m_{i,H,H'}]$ for an arbitrary $i \in H \cap H'$.
  - Otherwise, players reconstruct $a_H, b_H$, and let $[m_{H,H'}] = L(a_H b_H, 0)$

**Adding the terms**
- Players locally compute and output $[c] = \sum [m_{H,H'}]$. 
Shamir-Based VSS

- How can honest players guarantee that their shares are consistent?
  - Can’t exchange shares as in CNF-based VSS
  - Idea: “grill” the dealer via two-level sharing

\[ f_2(x) = 4x + 1 \]
\[ g_2(y) = 2y + 5 \]
given to P_2

\[ F(i,j) = s_{ij} \]

primary shares \( s_i \)

secondary shares
Example: $t=1$, $n=5$

- Dealer picks a random bivariate polynomial $F(x,y)=s+rx+ry+ryxy$ and sends to each player $P_i$ the polynomials $f_i(x)=F(x,i)$ and $g_i(y)=F(i,y)$.
- Each $P_i$ privately sends to each $P_j$ the common secondary shares $s_{ji}=f_i(j)$, $s_{ij}=g_i(j)$. $P_j$ compare these values to $s_{ji}=g_j(i)$, $s_{ij}=f_j(i)$ (respectively) and if they don’t match it broadcasts an inconsistency complaint $\{i,j\}$.
- Let $C$ be a “good clique” of size 4: a set of 4 players which are not involved in any complaint. If no such $C$ exists, dealer is disqualified.
- Each player $i \in C$ lets $s_i=g_i(0)$
- The player $j \not\in C$ recovers $s_j$ by applying error-correction to the 4 secondary shares $s_{ji}^i$ received from $i \in C$. 

```
 8 15 X2 29 36
 7 13 19 25 31
 6 11 16 21 26
 5 9 13 17 21
 4 7 10 13 16

s=1
 3 5 7 9 11
```
General Protocol, $n>4t$

- Dealer picks a random bivariate polynomial $F(x,y)$ with degree $\leq t$ in each variable and free coefficient $s$. It sends to each player $P_i$ the polynomials $f_i(x)=F(x,i)$ and $g_i(y)=F(i,y)$.
- Each $P_i$ privately sends to each $P_j$ the common secondary shares $s^{ij}_i=f_i(j)$, $s^{ji}_i=g_i(j)$. $P_j$ compare these values to $s^{ji}_j=g_j(i)$, $s^{ij}_j=f_j(i)$ (respectively) and if they don’t match it broadcasts an inconsistency complaint $\{i,j\}$.
- Let $C$ be a “good clique” of size $3t+1$: a set of $3t+1$ players which are not involved in any complaint. If no such $C$ exists, dealer is disqualified.
- Each player $i \in C$ lets $s_i=g_i(0)$
- The player $j \notin C$ recovers $s_j$ by applying error-correction to the $3t+1$ secondary shares $s^{ij}_j$ received from $i \in C$.

- Problem: How to efficiently find a good clique?
  - Noninteractive solution I: if $n>5t$, then a factor-2 approximation algorithm for vertex cover can be used to efficiently find a clique of size $n-2t \geq 3t+1$ when the graph contains a clique of size $n-t$.
    - If no such clique is found, the dealer can be safely disqualified.
  - Noninteractive solution II: more sophisticated version of the above; works efficiently when $n>4t$.
  - Interactive solution: dealer helps find a good clique by resolving disputes.
General Protocol, $n>3t$

- Dealer picks a random bivariate polynomial $F(x,y)$ with degree $\leq t$ in each variable and free coefficient $s$. It sends to each player $P_i$ the polynomials $f_i(x)=F(x,i)$ and $g_i(y)=F(i,y)$.
- Each $P_i$ privately sends to each $P_j$ the common secondary shares $s^i_{ji}=f_i(j)$, $s^i_{ij}=g_i(j)$. $P_j$ compare these values to $s^j_{ji}=g_j(i)$, $s^j_{ij}=f_j(i)$ (respectively) and if they don’t match it broadcasts an inconsistency complaint $\{i,j\}$.
- Dealer responds to each complaint by broadcasting the correct value.
- Players whose original shares contradict the broadcasted values broadcast an accusation against the dealer and halt.
- Dealer must broadcast the polynomials $f_i, g_i$ of all accusing players.
- Players who find new inconsistency accuse the dealer and halt.
- If there are more than $t$ accusations, dealer is disqualified. Otherwise players take their primary shares from the most recent version of $f_i$. 
Reducing $\text{MS}_{\text{Shamir}}$ to $\text{VSS}_{\text{Shamir}}$

- In fact: we reduce $\text{MS}_{\text{Bivariate}}$ to $\text{VSS}_{\text{Bivariate}}$, where “Bivariate” scheme shares $s$ using a bivariate degree-$t$ polynomial
  - For each primary share $s_i$ players hold secondary shares $[s_i]$ defined by a univariate polynomial $g_i$, where $g_i$ is known to $P_i$.
- **Observation**: all previous protocols for $\text{VSS}_{\text{Shamir}}$ can be easily modified to implement $\text{VSS}_{\text{Bivariate}}$
  - Instead of recovering its primary share $s_i$ alone, player $P_i$ can recover the polynomials $f_i, g_i$.

High level MS protocol:

- **Inputs**: Bivariate shares $[a],[b]$ (with secondary shares $g_{ai}, g_{bi}$)
- **Outputs**: Bivariate shares $[c]$ of $c=ab$
- Each $P_i$ computes $c_i=a_ib_i$ and applies $\text{VSS}_{\text{Shamir}}$ to $c_i$ resulting in $[c_i]$
- $P_i$ proves in “zero-knowledge” to the other players that the value it shared satisfies $c_i=a_ib_i$.
  - If proof fails, his shares $a_i,b_i$ are publicly reconstructed from their secondary shares, and $[c_i]=[a_ib_i]$ can be locally computed.
- Players compute $[c]$ as a proper linear combination of $2t+1$ of the $[c_i]$
  - Each $\mu_i c_i$ is reshared using $\text{VSS}_{\text{Bivariate}}$ by applying a linear function to $\mu_i c_i$ and randomness VSSed by $P_i$. 
Proving that $c_i = a_i b_i$ ($n > 4t$):

- **Inputs:** Shamir shares $[u], [v], [w]$, where sharing polynomials $g_u, g_v, g_w$ are known to $P_i$.
  - Here $u = a_i$, $v = b_i$, $w = c_i$
- **Outputs:** Each $P_j$ accepts if $w = u v$ and rejects otherwise.
- $P_i$ computes $g_{uv} = g_u g_v$ and applies VSS$_{Shamir}$ to each of the $2t+1$ coefficients of $g_{uv}$ except the free coefficient
  - The free coefficient is supposedly shared by $[w]$
  - Players need to verify that the degree-$2t$ polynomial $g_{uv}$ uniquely defined by the VSSed coefficients is indeed equal to $g_u g_v$
- For each $j$, the value $g_{uv}(j)$ is privately communicated to $P_j$ by having each player send to $P_j$ the appropriate linear combination of the shared coefficients.
- $P_j$ verifies that $g_{uv}(j) = u_j v_j (= g_u(j) g_v(j))$, otherwise it broadcasts a complaint.
- Players accept iff there are at most $t$ complaints.

- **Privacy:** $g_{uv}(j)$ is already known to $P_j$
- **Completeness:** If $P_i$ is honest and $w = u v$, then no honest player will complain.
- **Soundness:** If there are $\leq t$ complaints then $g_{uv}(j) = g_u(j) g_v(j)$ for $\geq n-2t > 2t$ points $j$.
  Since $\deg g_{uv} = \deg g_u g_v \leq 2t$, we must have $g_{uv} = g_u g_v$ and hence $w = u v$. 
Handling $n > 3t$

- **Idea:** resolve a complaint of $P_j$ by confronting him with $P_i$.
- **Problem:** $u_j, v_j$ are only known to $P_i, P_j$.
- **Solution:**
  - Convert the input shares to bivariate shares
    - Similarly to last step in MS protocol
  - In case of complaint by $P_j$, the values $u_j, v_j, g_{uv}(j)$ are publicly reconstructed.
    - If $u_j v_j \neq g_{uv}(j)$, players reject.
Computational Security
Overview

• **Goal**: Obtain computational security against an active adversary.

• **Hope**: under a reasonable cryptographic assumption, obtain a computationally $n$-secure protocol for arbitrary functionalities.
  – Impossible in information-theoretic setting.
  – Possible* in computational setting.
  – In fact, assuming one-way functions exist, there is a *general compiler* transforming an $n$-private protocol into an $n$-secure one with “similar” efficiency.

$\Rightarrow$ General $n$-secure computation is possible if passively secure OT exists.
  • Implied by rerandomizable encryption, or trapdoor permutations.
How to Flip a Coin?

• Information theoretic setting
  – Perfect coin-flipping is possible iff $n > 3t$.
  – Assuming broadcast, statistical coin-flipping is possible iff $n > 2t$.
  – Nothing can be done in the 2-party case.

• Idea for (real-life) 2-party coin-flipping:
  – $P_1$ writes a random bit $b$ and locks the paper in a box.
  – $P_2$ picks a random bit $b'$
  – $P_1$ unlocks box and reveals $b$
  – Players output $b \oplus b'$
Is the protocol secure?

• Idea for 2-party coin-flipping:
  – \( P_1 \) writes a random bit \( b \) and locks the paper in a box.
  – \( P_2 \) picks a random bit \( b' \)
  – \( P_1 \) unlocks box and reveals \( b \)
  – Players output \( b \oplus b' \)

• Properties of coin-flipping in the ideal model:
  – If both players are honest, they output the same (random) bit
  – The bit output by an honest player is random
    • In particular, corrupted player cannot bias the output of the other player.

• Problem with protocol: what if \( P_1 \) refuses to open box?
  – Suggestion: \( P_2 \) picks a random bit as its output.
    • Problem: Allows \( P_1 \) to get the value it wants with probability 3/4
  – Real-life suggestion: \( P_2 \) picks the output value that would make \( P_1 \) most unhappy.
    • Allows coin-flipping where each player can only bias the coin to one direction.
    • Applicable in many scenarios (e.g., in games, gambling).
    • Problem: not always clear what would make \( P_1 \) unhappy…
  – Convention: \( P_2 \) outputs \( \bot \) (i.e., does not learn the output).
Can the problem be fixed?

• No, problem is inherent.
• High-level argument:
  – Suppose protocol has $m$ rounds, where players alternate in sending messages.
  – In honest execution, at some stage some player should have an advantage of $1/m$ over the other in the probability of guessing the output.
  – An adversary corrupting this player can decide whether to continue the protocol or to abort based on its guess of the output.
• Workarounds:
  – Relax security definition allowing adversary to abort in the ideal model as well.
  – Disperse adversary’s advantage over many rounds.
  – Enforce penalties for aborting (e.g., using bitcoin infrastructure)
  – Timing assumptions:
    • $P_1$ uses a weak lock that can be opened via brute force within an hour.
    • $P_2$ must respond with his guess $b'$ within 1 second.
    • If $P_1$ fails to open box, $P_2$ can recover the locked value $b$ within an hour and output $b \oplus b'$. 
Fairness

• Our default definition for secure computation insists on total fairness: in the ideal model, all parties get their outputs from TP.
• As indicated by coin-flipping example, total fairness cannot always be achieved.
• Motivates the following relaxed security definition: in ideal model, TP first sends outputs to simulator, which then decides whether to continue (letting TP send the actual outputs to uncorrupted parties) or to abort (letting TP send a special symbol ⊥ to uncorrupted parties).
  – Better: simulator can abort only if it corrupts P₁
    • Allows fully secure computation of any asymmetric 2-party functionality: functionality in which only P₁ gets an output.
  – Under abovedefs., every functionality can be computed with computational n-security.
    • Assumes broadcast or public-key infrastructure.
Back to Coin Flipping

• Main tool: **Commitment** scheme
  – Cryptographic analogue of a locked box

• **Def.** A (non-interactive bit-) commitment scheme is an efficiently computable function $\text{Com}(b,r)$ with the following properties:
  – **Privacy:** $C_0(k) \approx_c C_1(k)$, where $C_b(k)$ is the distribution of $\text{Com}(b,r)$ with $r \in \mathbb{R}\{0,1\}^k$.
  – **Binding:** there are no $r_0, r_1$ such that $\text{Com}(0,r_0) = \text{Com}(1,r_1)$.

• **Features:**
  – Receiver cannot learn committed value without help of sender.
  – An honest sender can efficiently “open” commitment by sending $b,r$.
  – Even a malicious sender cannot “change its mind” once committed.
Implementing Commitment

• GM encryption is committing
  – Com(b,r) = GM public-key + encryption of b using randomness r
  – Technical issue: what if r fails to produce a valid key+encryption pair?
    • Happens w/neg. probability, so in such case can output b without violating secrecy.

• Commitment from 1-1 OWF
  – A one-way function (OWF) is an efficiently computable f:{0,1}∗→{0,1}∗ that is “hard to invert” in the following average-case sense: for any efficient (nonuniform) inversion algorithm I, the probability that I(1^k,f(r)) returns a preimage of f(r), where r∈R{0,1}^k, is negligible in k.
  – Given a 1-1 (bijective) OWF f, a commitment may be defined as:
    • Com(b,(r_1,r_2))=(f(r_1), r_2, b⊕<r_1,r_2>), where < , > denotes mod-2 inner product.

• Interactive commitment is possible under the minimal assumption that (an arbitrary) one-way function exist.
Coin Flipping Protocol

$P_1$      $P_2$

$b \in \mathbb{R}\{0,1\}$

$r \in \mathbb{R}\{0,1\}^k$

$c \leftarrow \text{com}(b,r)$

$c$

$b' \in \mathbb{R}\{0,1\}$

$b', r$

Output $b \oplus b'$

Output $b \oplus b'$ if $c = \text{Com}(b,r)$

Output $\bot$ otherwise
Simulating $P_2^*$

- Call TP and receive a random bit $c$.
- Repeatedly run the protocol $(P_1, P_2^*)$ until $P_1$ outputs $c$.
  - If $k$ attempts fail, output “Fail” and halt.
- Output the view of $P_2^*$ in the last (successful) run.

- **Claim**: Above simulator *statistically* emulates the real process.
  - By the privacy of the commitment scheme, the distribution of $c$ in the real world is statistically close to uniform.
  - The probability that the simulator outputs “Fail” is negligible.
  - Given that the simulator does not output “Fail”, its output is distributed identically to the real view of $P_2^*$ given that $P_1$ outputs $c$.

- **Question**: Does the above simulator generalize to flipping many coins in parallel?
Simulating $P_1^*$

- Call TP and receive a random bit $c$.
  - At this stage, simulator still can’t tell whether to abort.
- Start running $P_1^*$ and obtain a commitment $c^*$ it would send in Round 1.
- Continue running the protocol from this point in two different ways, using $b'=0$ and $b'=1$. Let $c_0,c_1$ be the resulting outputs of $P_2$ (where $c_b \in \{0,1,\perp\}$).
  - If $c_0 = c_1 = \perp$, abort TP and output a random $b'$ as incoming message.
  - If $c_d \neq \perp$ and $c_{1-d} = c'$
    - If $c = c'$, output $1-d$ as incoming message and allow $P_2$ to receive its output $c$.
    - Otherwise, abort TP and output $d$ as incoming message.
  - If $c_d \neq c$ and $c_{1-d} = 1-c$, output $d$ as incoming message and allow $P_2$ to receive its output $c$.

- **Claim**: Above simulator perfectly emulates the real process.
Zero-Knowledge Proofs

• Let $L$ be an NP language and $R$ a corresponding NP-relation.
• A Zero-Knowledge Proof system for $L$ (using $R$) is a protocol between Prover $P$ and Verifier $V$, satisfying the following.
  – **Inputs**: $P$ and $V$ have a common input $x$, and $P$ holds a potential witness $w$.
  – **Outputs**: $V$ either accepts or rejects. $P$ has no output.
  – **Completeness**: If $(P,V)$ run on inputs $(x,w) \in R$, then $V$ accepts $x$.
    • Perhaps except w/neg prob.
  – **Soundness**: If $x \not\in L$ then for any malicious prover $P^*$ and any $w$, the probability that $P^*(x,w)$ makes $V$ accept $x$ is negligible in $|x|$.
    • By default, $P^*$ can be computationally unbounded. ZK Proofs in which $P^*$ is bounded are referred to as ZK arguments.
  – **Zero-knowledge**: For any efficient malicious verifier $V^*$, there is an efficient simulator such that for all $(x,w)$ the output of $S(x)$ is computationally indistinguishable from the view of $V^*$ in its interaction with $P$ on inputs $(x,w)$.
    • Perfect and statistical variants
Zero-Knowledge Proofs (contd.)

• Thm. For every NP-relation $R$ there is a zero-knowledge proof system.

• In fact, most natural ZK proofs systems satisfy the stronger requirement of zero-knowledge proofs of knowledge.
  – Intuitively: Verifier is convinced that whenever it accepts $x$, the malicious prover $P^*$ “knows” some witness $w$ such that $(x,w) \in R$.
  – A bit more formally: There exists an efficient extraction algorithm $E$ and negligible function $\varepsilon$ such that whenever $P^*$ makes $V$ accept $x$ with prob. $> \varepsilon(|x|)$, $E$ can use $P^*$ as a subroutine to extract a valid witness $w$ for $x$. 
From Passive Security to Active Security

- GMW compiler, in a nutshell:
  - Use commitment to make each player commit to its input.
  - Use committed coin-flipping to make each player commit to a secret and uniformly chosen random input for the private protocol.
  - Run the private protocol, where for each broadcasted message its sender proves in zero-knowledge that his current message is consistent with his committed input, committed random input, and previous messages.
    - The latter is an NP-statement, which can be efficiently verified given the randomness used for committing the inputs and random inputs.
    - If such proof fails, honest players detect cheating and can abort with output ⊥.

- Some subtle issues remain.
  - For instance
    - How does one make sure that the input committed to by $P_2$ is not related to the input committed to by $P_1$? (Malleability)
    - How does the simulator “extract” the inputs it should send to TP?
    - Both of the above issues are addressed by making each player prove it knows the input it committed to using zero-knowledge proofs of knowledge.
What’s left to be done?

• Improve efficiency
  – Many models, many problems
  – Various efficiency measures
    • Communication, rounds, local computation, randomness

• Better composability
  – Universal Composability is provably impossible in standard model (for most functionalities).
  – Goal: find useful relaxations of UC which can be realized.

• Stronger adversaries
  – Better protocols for adaptive/mobile adversaries
  – Better/different solutions to the fairness problem

• Weaker cryptographic assumptions
  – MPC is equivalent to OT
  – Goal: Generalize / Unify assumptions under which OT can be based.

• Obtain lower bounds and negative results.